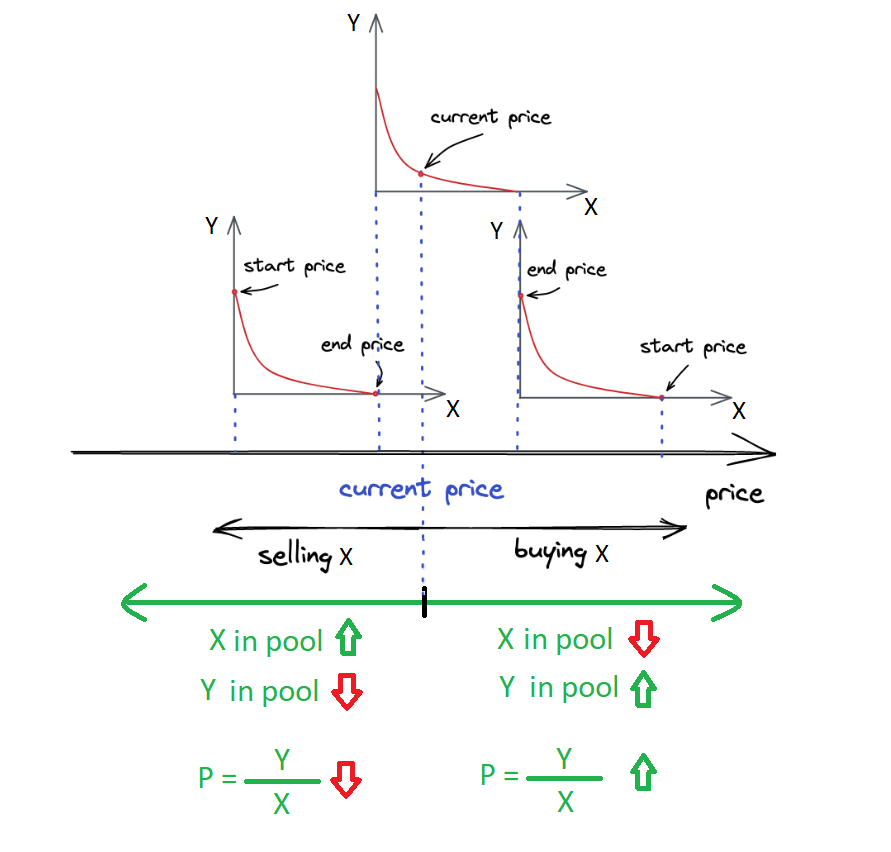
Company: SayNode

How are X and Y calculated?

There are 3 different situations that determine how the x and y are calculated These depend on the upper (Pu) and lower (Pl) prices that define the range in which the liquidity provider wishes to add liquidity, and the current price (Pc) the pool is currently at.

This is because the V3 is not a normal AMM. It has virtual and real liquidities. So, if it has a certain number of tokens within 2 price ranges and someone makes a swap big enough for the price to exit those price limits, it needs to grab extra tokens to complete the swap (token1 or token2 depending on the swap being made). This is why only one of the tokens is added when we want to add liquidity outside of the current range: the pool assumes it only needs the tokens that are needed to complete a swap in that direction.



Let us imagine that the current price of the X/Y pool is Pc=5:

1. **The user wants to provide liquidity in a range below the current price (Pu<Pc), for example Pl=1 and Pu=4.**

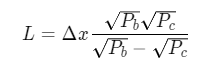
In this situation, only Y tokens will be added to the liquidity, so we only calculate one liquidity and there is no need to recalculate the Y amount or the X amount (X=0), although the contract will recalculate it anyway (which results in a certain slippage but expected).





1. **The user wants to provide liquidity in a range that includes the current price (Pl<Pc<Pu), for example Pl=4 and Pu=6.**

In this situation (the most usual) both tokens will be sent to the contract. The contract receives the intended amount of x or y, and calculates the corresponding liquidity,

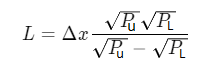
then recalculates the x and y token amounts

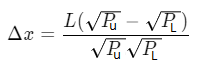
 

In this case Pa=Pl and Pb=Pu.

1. **The user wants to provide liquidity in a range above the current price (Pl>Pc), for example Pl=6 and Pu=10.**

In this situation, only X tokens will be added to the liquidity, so we only calculate one liquidity and there is no need to recalculate the X amount or the Y amount (Y=0), although the contract will recalculate it anyway





Assumptions:

Since we need to send a liquidity to the contract, this means we first need to calculate it in the frontend. Since different languages have different rounding systems and decimal amounts, the calculations in the frontend and in the contract are slightly different.

We allowed a 1% difference between the frontend and the contract calcs. This has no affect in the contract logic or in the “fairness” of the amounts the user deposits or swaps, it just means that the predicted values that are shown in the frontend are 1% different from what it will be sent/received. This is the best way we can show the user aproximatly what he will send/receive without him needing to waste Tez to get an exact calculation from the contract.

What will change to go live?

* At the moment we can only interact with a pool of some FA2 tokens we deployed (X and Y). If we go live we will add a “Create Pool” button so the users can create their own pools and interact with them (they are added to the swap frontend);
* Get token symbols for the frontend (we need some clarification of what is the best place/way to get these);
* Get the current earned fees for each position;
* Add a change position option;
* Visual upgrades;

Wallet seed (test wallet):

Private key: edskRsnyieXbHqMXZYTNKaod9QgKq4bQZkKy8YRgSH8ixdY6cLerEooVAXamJQAf77Rq6WoEuNKJzsqGC8HWr9CWiyhEeoNxaL

Seed phrase: chunk coffee deliver dad bunker rally thunder toe exhibit door squirrel puzzle

If you want to test with a random wallet, simply let me ([joao@saynode.ch](mailto:joao@saynode.ch)) know or Simon, and I will send you tokens.